

AD A 024819

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Note No. 68	2. GOVT ACCESSION NO. (14) TN-68	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ARPA Internetwork Protocols Project Status Report	5. TYPE OF REPORT & PERIOD COVERED (9) Technical Note	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Vinton G. Cerf	8. CONTRACT OR GRANT NUMBER(s) MDA903-76C-0093, NEW ARPA Order 2494	9. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 6T10
9. PERFORMING ORGANIZATION NAME AND ADDRESS Stanford Electronics Laboratories Stanford University Stanford, CA 94305	10. REPORT DATE November 15, 1975	11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency Information Processing Techniques Office 1400 Wilson Ave., Arlington, VA 22209
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency Information Processing Techniques Office 1400 Wilson Ave., Arlington, VA 22209	12. REPORT DATE November 15, 1975	13. NUMBER OF PAGES 12 25 p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Mr. Philip Surra, Resident Representative Office of Naval Research Durand 165 Stanford University	15. SECURITY CLASS. (of this report) Unclassified	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Reproduction in whole or in part is permitted for any purpose of the U. S. Government "A"		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) <div style="border: 1px solid black; padding: 5px; display: inline-block;"> DISTRIBUTION STATEMENT A Approved for public release; Distribution Unlimited </div>		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) PACKET NETWORK, INTERCONNECTION, PERFORMANCE EVALUATION, COMMUNICATION PROTOCOLS, PACKET RADIO NETWORK, ARPANET was implemented		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) We have implemented an Internetwork Communication Protocol which permits reliable interprocess communication to be established between hosts in different networks. Estimates of program size are made for a single connection version of the program for use in a mobile packet radio terminal. Preliminary throughput measurements indicate serious problems with the Very Distant Host interface and its related software. A plan for detailed analysis of delays in the system has been made and the results will be reported during the next quarter.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408 071

mt

**Best
Available
Copy**

ACCESSION FOR	
NTIS	DDP Section <input checked="" type="checkbox"/>
DDC	DDP Section <input type="checkbox"/>
UNCLASSIFIED	<input type="checkbox"/>
JUSTIFICATION	
<i>Per ltr</i>	
BY	
DISTRIBUTION AVAILABILITY CODES	
Dist.	UNCLASSIFIED
<i>A</i>	

ARPA INTERNETWORK PROTOCOLS PROJECT

STATUS REPORT

by

Vinton G. Cerf
Principal Investigator
November 15, 1975

Technical Note #68

DIGITAL SYSTEMS LABORATORY

Dept. of Electrical Engineering Dept. of Computer Science
Stanford University
Stanford, California

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency or the United States Government

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2494, Contract No. MDA903-76C-0093

ARPA INTERNETWORK PROTOCOLS PROJECT
STATUS REPORT

Vinton G. Cerf
Principal Investigator
November 15, 1975

1. TCP Implementation

At the date of this writing, the Transmission Control Program (TCP) at Stanford University - Digital Systems Laboratory (SU-DSL) was complete, except for

- (a) code to perform desynchronization
- (b) code to handle the arrival of FIN messages when the TCP is in a state other than ESTABLISHED. A proposal has been circulated to the other participants in the internetwork experiments but comments have not yet been received.

1.1 New CLOSE Procedure

During the period, a change was made to the connection closing mechanism which required reprogramming of the CLOSE code as well as the TCP (Transmission Control Block) deletion code, since TCB's can now only be deleted when the user has CLOSED the connection and FIN's have been exchanged AND acknowledged. The latter feature is the new procedure and guarantees that the initiator of the CLOSE will be able to distinguish between the case that the FIN was received normally at the destination and the case that the connection has somehow been terminated abnormally at the remote side. In the earlier design, it was sufficient to send and receive a FIN and to get a local CLOSE from the user. However, a FIN send in response to a FIN received might not in fact arrive, and the retransmission of the originating FIN would then get a "connection non-existent" message in return, leaving the initiator of the close uncertain whether his FIN had arrived or not.

1.2 TCP size and organization

The SU-DSL TCP is written to run under an ELF operating system kernel. The equipment at SU-DSL is shown in figure 1. The basic software organization is shown in figure 2 and the approximate sizes of the various TCP software components shown in Table 1. Initially, most of the TCP was programmed in BCPL (Basic Compatible Programming Language).

Each module in Figure 2 is a separate ELF process running independently. The user calls are serviced by the user call interface code, reached by Emulator Traps (EMT's). The other processes communicate among themselves by sending and receiving signals.

As can be seen in Table 1, the size of the TCP leaves little room in the 28k word memory of the PDP-11/20 for experimental user software. A part of the problem is the cost of the BCPL high level language. To remedy some of the space difficulties, we plan to reprogram parts of the TCP in MACN11 assembly language. The generality of the implementation (arbitrary number of connections, dynamic buffer allocation, etc.) has an undeniable space cost (see section 1.3 below on Packet Radio TCP-0 implementation). The lowest level service routines, heretofore written in ELF are being reprogrammed in assembly language with space reductions up to 70% in some cases.

1.3 Single Connection TCP for Packet Radio Terminal

We have ordered an LSI-11/03 with 8k words of memory, a 16 bit parallel interface and a terminal interface. We plan to build an interface which conforms to BBN-1822 for the purpose of attaching the terminal to a Packet Radio unit (PRU). Initially, we will use an LSI-11/03 at SRI to test our TCP/TELNET code, until we have a PRU of our own delivered to SU-DSL. The PRU will be used both to connect the PDP-11/03 terminal to the PRNET (Packet Radio Net) and, alternatively to connect our PDP-11/20 as a host. Eventually,

we hope to test terminal, host, and station concurrently. Figure 3 illustrates the terminal hardware and figure 4 (and table 2) the software plan.

We are hoping to fit the entire terminal software package in 4k words, but are not sure how much code the host/PRNET interface will require in the way of support software. In any case, the TCP, which only runs a single, full-duplex connection, should not require more than 2.5k words, handcoded in assembly language (MACN11).

A preliminary software specification for TCP-0 is to be delivered by 15 November and a final specification with implementation and documentation complete by 1 February 1976.

1.4 PRNET Host software/hardware

In addition to the PRNET terminal, we have installed an IMP-11A interface for our PDP-11/20 and are awaiting delivery and installation of the PRU. Initially, this unit will be installed in the SU-DSL machine room with its antenna on the roof. We have initiated work to pull cables to the roof of the 4 story Durand building nearby and plan to attach the PRU antenna to one of the existing Durand microwave towers. Sufficient rack space for the PR repeater has been allocated in the Instructional TV Facility equipment room and an 11-15 pair cable will be run from the roof of Durand to its basement, as shown in figure 5. The high antenna should give excellent range to the PRU (e.g. to SRI and Eichlerville units). Installation of the PRU is expected in the middle of February 1976 (horseback estimate).

Software for the PDP-11/20 PRNET host will include ELF, a reduced TCP, PRNET/HOST software, IMP-11A driver, and various simple service routines (e.g. server TELNET and some as yet unspecified interactive application programs). If the general TCP proves too large, we can demonstrate the host

using TCP-0, but this would require the coding of a server TELNET to interface to TCP-0. We hope this won't be necessary.

2. Experiments

2.1 Internet packet exchanges

Connections have been established between SU-DSL/BBN and SU-DSL/UCL. Data has been exchanged. SU-DSL has always been the initiator since other sites do not yet have an exerciser (terminal controller - although BBN may have a primitive TELNET nearly completed). All sites (BBN, UCL, SU-DSL) have opened looped connections to themselves, but only BBN and SU-DSL have successfully CLOSED connections (UCL is writing CLOSE code and FIN handling now).

BBN has a functioning echoer and SU-DSL has opened connections to it, sent data, and closed the connection (a syntax error in SU-DSL code caused its TCP to crash before the TCB was actually removed, but this has been repaired).

2.2 Initial Performance Tests

So far as we know, the BBN PDP-10 implementation is still using JSYS traps and therefore runs at about 1 letter/second. We have been testing throughput by opening a connection to ourselves (send and receive ports are looped). As a result, we have begun to explore changes in implementation choices to alleviate some bottlenecks.

One of the goals of the TCP implementation was to allow for the "piggy-backing" of acknowledgments on return traffic. In the absence of such traffic, of course, empty packets with acknowledgment information must be forced out to avoid unnecessary retransmissions from the sender. In our initial implementation, new arriving packets cause a "NEWRECEIVE" flag to be set.

If packets are produced and sent out the reverse channel, the accompanying acknowledgment causes this flag to be reset. The retransmission process maintains a queue of wake-up times (one per connection) arranged in order, nearest event first. When it awakens, it checks to see if a "NEWRECEIVE" flag is set for this connection and sends out an empty acknowledgment before resetting the flag and goes on to check whether any packets still on the retransmission queue should be retransmitted. We chose not to modify the acknowledgment information in retransmitted packets since this would require modification of the packet checksum and might lead to serious confusion at the destination TCP when dealing with Gateway fragmentation. After servicing a connection, the retransmission routine determines the earliest time at which it should next awaken and schedules a signal for this time. If the earliest required time is greater than a default constant for the retransmission process (e.g. $\frac{1}{2}$ - $\frac{1}{2}$ second), then the process wakes itself after the shorter interval. This arrangement can lead to some odd interactions when sending the output of a full duplex connection directly into the input side, as shown in figure 6.

In the self loop experiment, a connection is established between the send and receive sides of the same port. Letters are sent and received and various statistics collected to highlight the behavior of the TCP.

2.2.1 Single letter at a time throughput

A user program was written (modification of the conventional exerciser) which would send a letter of a fixed length, and wait until it was acknowledged at which time a second letter was sent. This corresponds roughly to RFNM (request for next message) driven NCP experiments.

Letter length (i.e. actual text length in octets) was varied as shown in Table 3. The packet retransmission timeout was synonymous with the retransmission process' default wake-up timeout.

The results of this experiment clearly show that when no reverse traffic is waiting (i.e. in this case, only one message at a time is being sent), then all acknowledgments are sent by the retransmission process. Furthermore, the number of unnecessary retransmissions increases as the retransmission process wake up time decreases (and packet retransmission time decreases correspondingly). In fact, for the 0.25 second retransmission timeout, there were sometimes more retransmissions than letters sent. Two problems were evident, first, the retransmission process wake-up time was unnecessarily tied to the packet retransmission time and second, ACKs were not getting to the source fast enough. Of course, in this self loop, there may be some interaction among the send/receive sides of the TCP operating on the same port (e.g. locks for TCB state information, priority of TCP and user process, etc.) which would not ordinarily be found in a connection to a port at a different TCP. In this case, when the retransmission process awakened, it realized that a packet had to be retransmitted and an ACK had to be sent. Thus at least one retransmission always went with the ACK. As the retransmission timeout decreased, the situation accentuated itself noticeably.

To more precisely observe this behavior, we separated the packet retransmission timeout from the nominal retransmission process timeout. We also modified the TCP code to allow for one of three kinds of acknowledgment procedure, as shown below:

- (a) send an ACK immediately upon delivering a packet into a user buffer.
Set "NEWRECEIVE" flag whenever an acceptable packet (even a duplicate) arrives and reset this flag whenever an ACK is sent.
- (b) same as (a) for NEWRECEIVE, except set this flag on delivering data into a user buffer rather than forcing an ACK to be sent.

(c) same as for (a) as far as NEWRECEIVE flag treatment, but send an ACK on delivery of data to user buffer only when the send letter queue is empty, otherwise, set the NEWPECEIVE flag instead.

We have tested alternatives (a) and (b) with a single message at a time transmission regime, with the results shown in table 4. The results are not entirely consistent. When ACKs are not forced, and both timeouts are 0.5 seconds, we see that roughly 120(127 actually) letters and retransmissions were sent. This corresponds to one letter every 0.5 seconds. Letters are sent roughly as often as ACKs are generated every 1/2 second by the retransmission process. When ACKs are sent immediately upon delivery of new data (line 5, Table 4) the number of retransmissions drops to 8 and the number of letters jumps to 151. The strategy of waiting for the retransmission process to send ACKs (in the absence of reverse traffic) is clearly a poor one. As the retransmission process is awakened more frequently, to reduce the ACK delay, more and more bandwidth is used up with retransmissions (lines 1-4).

We can try to sketch the flow of events which account for the behavior we have observed. In figure 7, we show time advancing from the top of the figure towards the bottom. The left time line is for events occurring on the send side of the TCP while the right time line is for events on the receive side. Transmission of information back and forth is indicated by arrows sloping downward to show time delay. Taking, for example, line 5 of Table 4, we see that no ACKs were ever sent by the retransmission process and that only a few retransmissions occurred. We speculate that this is so because, most of the time, a letter (packet) is received, delivered to a user buffer, and an ACK forced out and received (thereby removing the packet from the retransmission queue) before the 0.5 second retransmission time expires.

In figure 8 we show how retransmission may occur for the line 5 (Table 4) case. Basically, the ACK is somehow delayed in delivery so that the retransmission process is able to schedule a retransmission. This is an alarming result because it implies that the round trip time for the message and its associated ACK can exceed 0.5 seconds. We will investigate this phenomenon (it is possible that the self-looping and interference between send/receive semaphoring on the same connection is the source of this odd behavior).

In figure 9, we illustrate how ACKs only or ACKs and retransmissions might occur, accounting for the statistics in line 1 of table 4.

It can be seen from line 2 of Table 4 that the number of ACKs-only increases to 117 when the retransmission process is run twice as often. This simply is the result of running the retransmission process before the packet has been delivered to the user. Line 3 of Table 4 shows increases in ACKs only and ACKs with retransmissions because the retransmission process appears to be catching the NEWRECEIVE flag both when set on packet arrival and when set because data has been delivered to the user.

It is apparent that a basic problem is that there is a long delay from the time a packet arrives until the time it is delivered to the user. When the retransmission timeout for a packet is 1/4 second, there is a substantial increase in the number of retransmissions. We will investigate this further and specifically measure the delays to find out where they are coming from. One conjecture is that the multi-process implementation uses substantial overhead in the ELF scheduler. This scheduler is run after most system calls (e.g. Signal, wait, P, V...) and could be using a large fraction of the CPU cycles. We can measure this and will report on it.

2.2.2 Multiple Letter Throughput

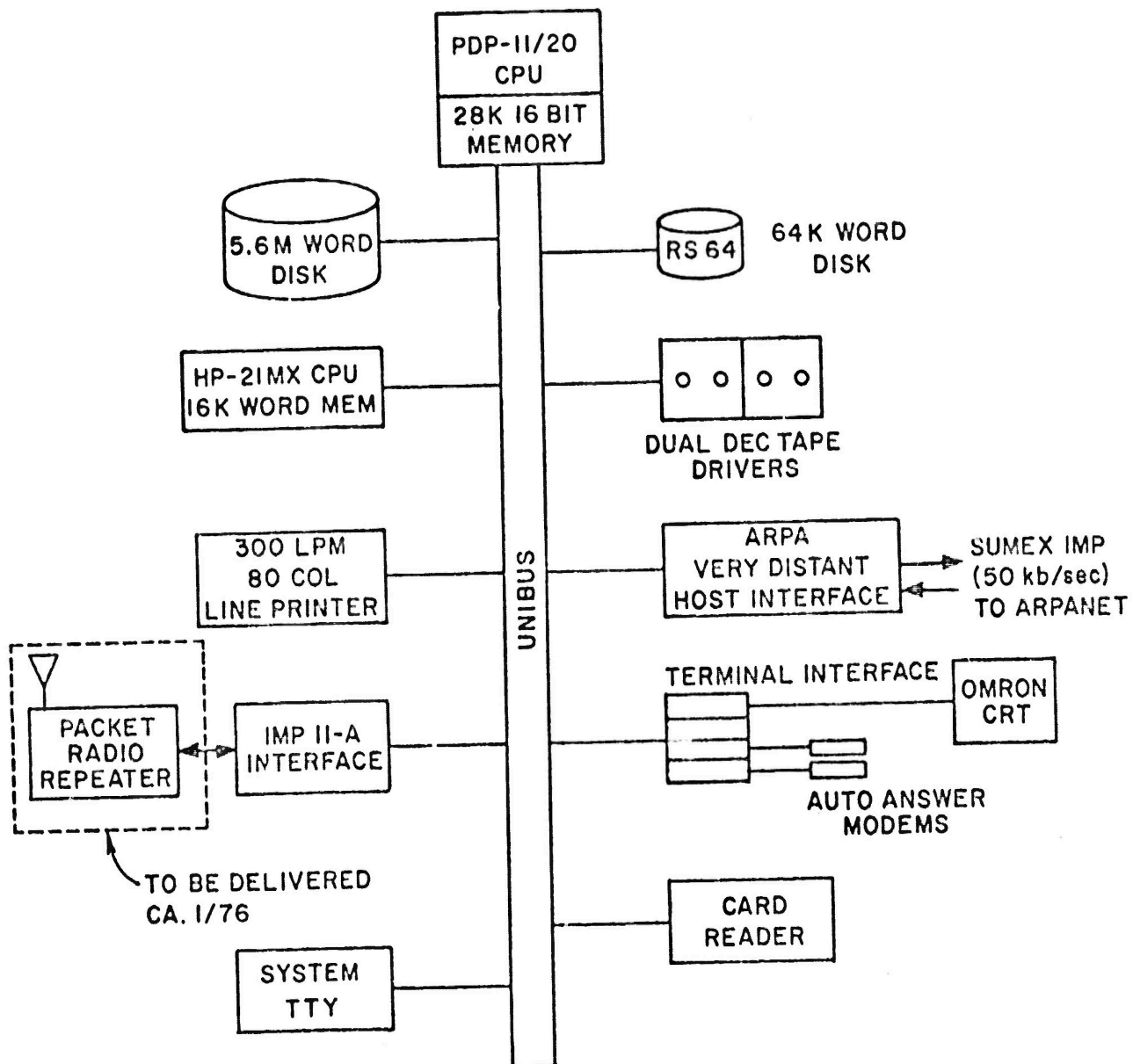
To determine what effect "filling the pipe" might have, we tried having two letters "outstanding" with the results as shown in Table 5.

With two letters outstanding there was an increase in throughput and, with 0.5 second retransmission time, the ratio of letters/retransmissions remained about 3:1. The number of empty ACKs sent dropped substantially since the second letter often carried the ACK for the first. However, disaster struck when the retransmission time was reduced to 0.25 seconds when 2/3 of all data transmissions were retransmissions and letters equalled ACKs in number.

The statistics of Table 5 come from an acknowledgment procedure which delays ACKs until the retransmission process times out or new data is sent. We expect better results when ACKs are forced out when data is delivered to the user.

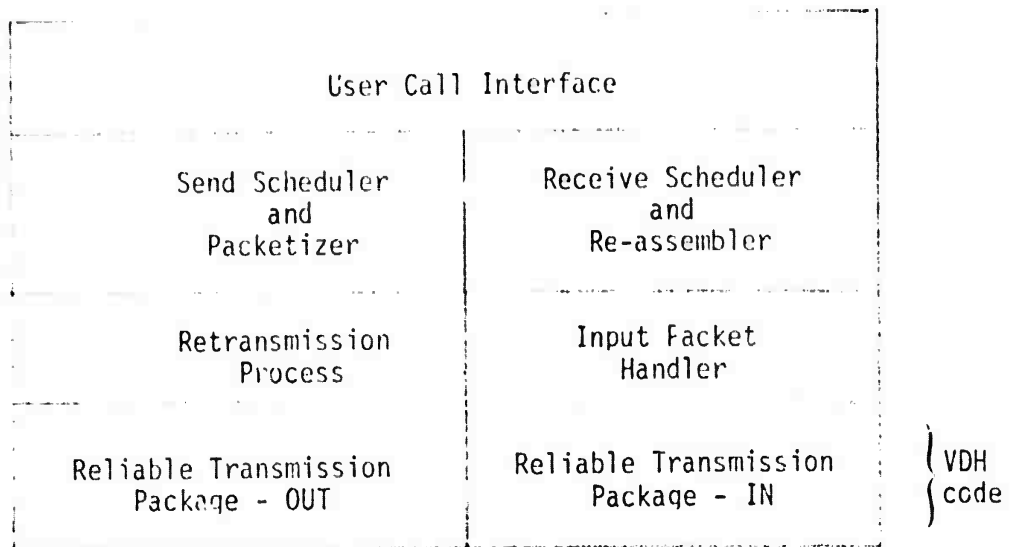
3.0 Security Work

In a separate report on secure packet fragmentation, we discussed a method for allowing encrypted packets to be fragmented at an internet gateway, but decrypted without reassembly.



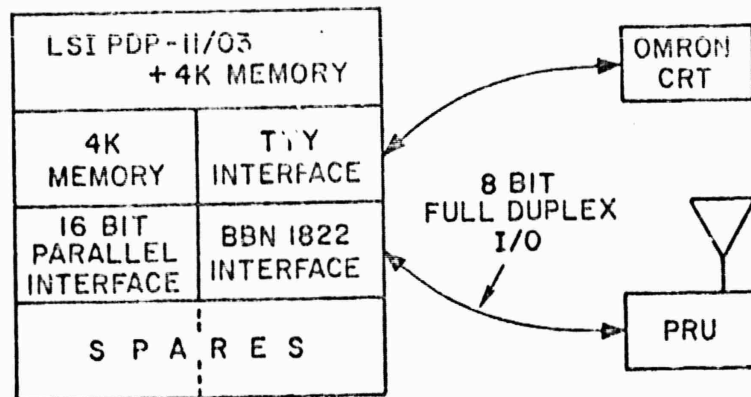
SU-DSL Equipment

Figure 1



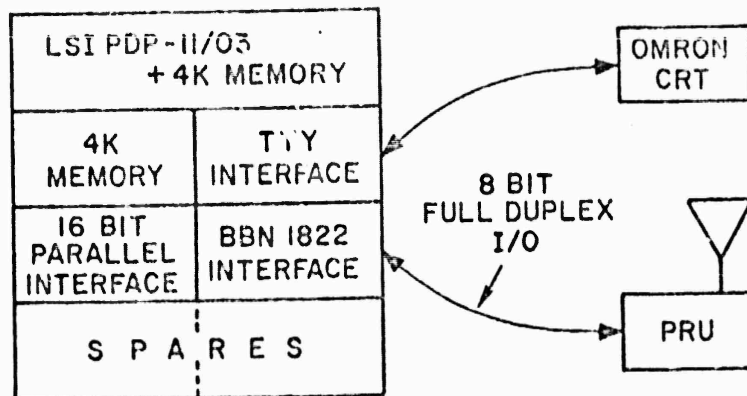
TCP Software Organization

Figure 2



LSI PDP-11/03 Chassis Organization

Figure 3



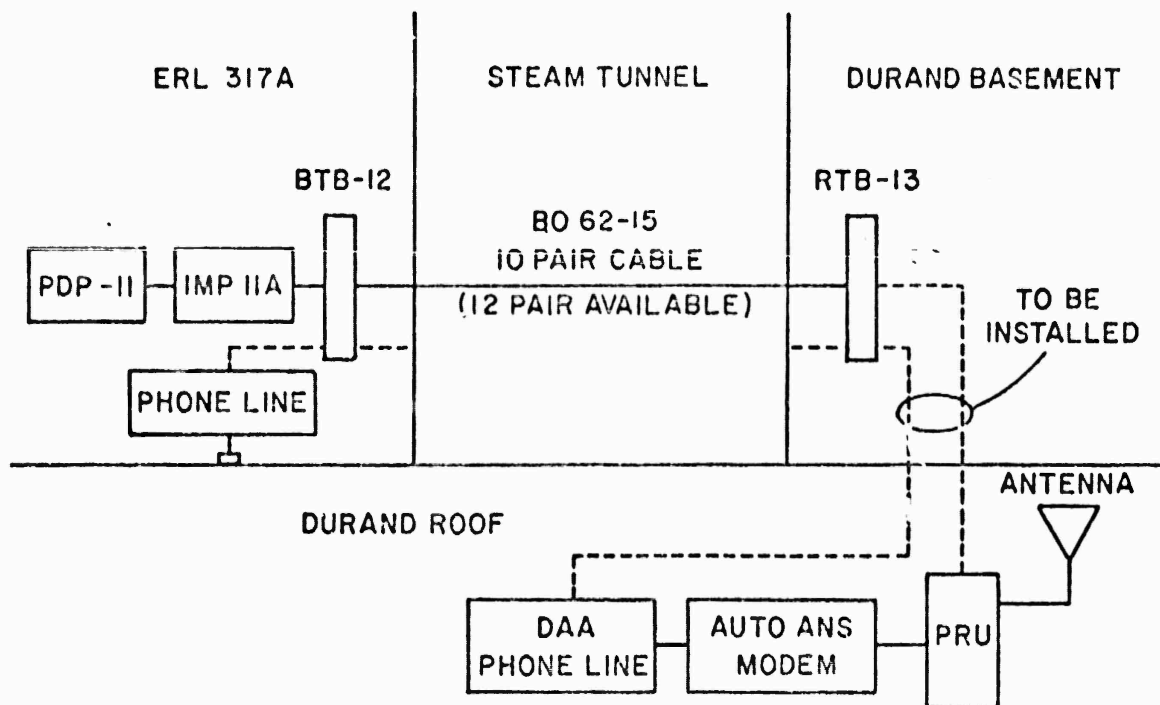
LSI PDP-11/03 Chassis Organization

Figure 3

Interrupt Dispatch	
HOST/PRNET Software	
TCP Send	TCP Receive
TELNET	
Terminal I/O	

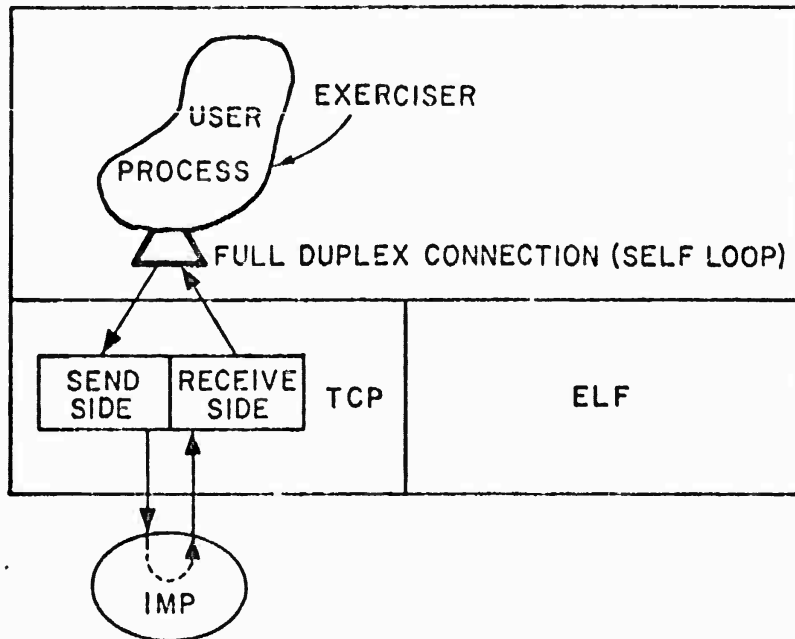
TCP-0 Software + Support Programs
for Packet Radio Terminal

Figure 4



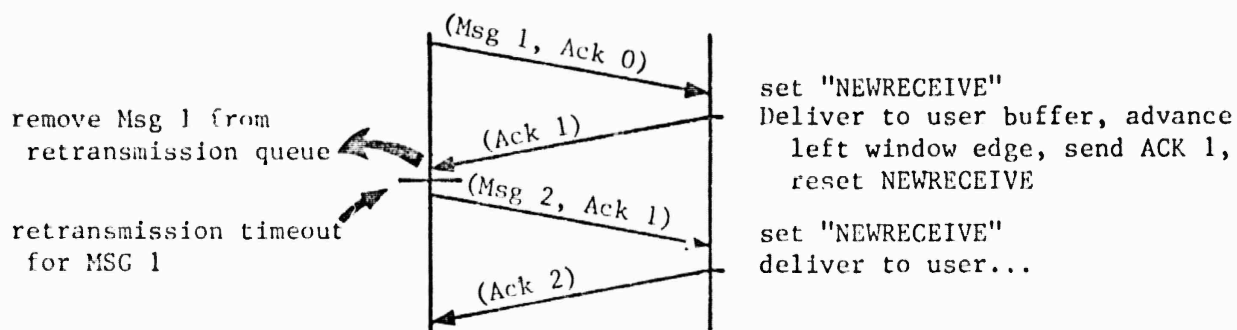
PDP-11/20 PRNET Host Equipment

Figure 5



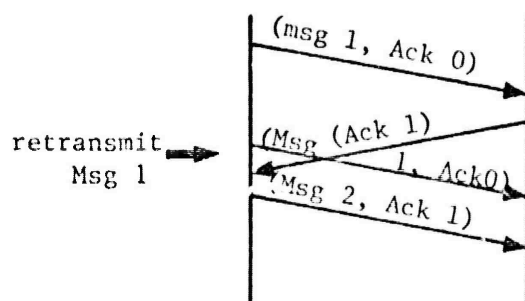
Self-loop SU-DSL Experiment

Figure 6



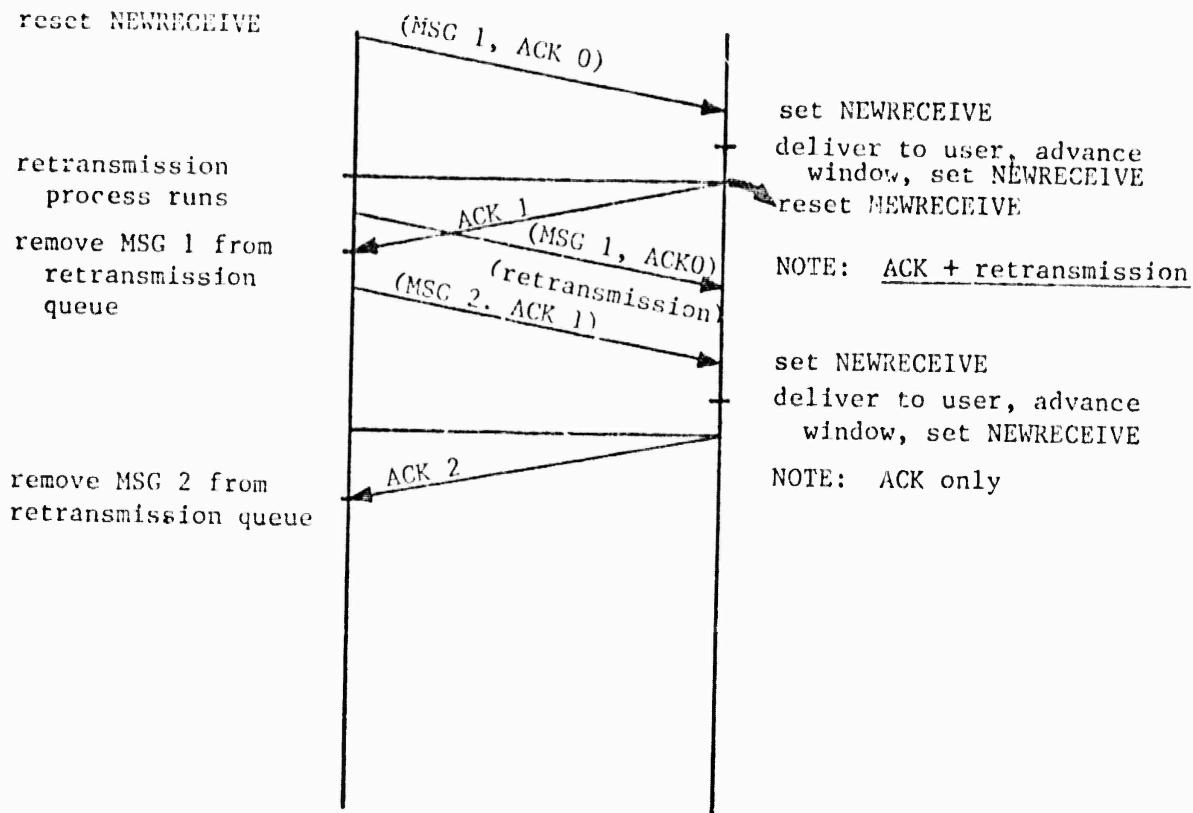
Forced ACK Time Diagram

Figure 7



Retransmission with Forced ACKs

Figure 8



Delayed ACK model

Figure 9

System Component		Size in 16 bit Words
ELF kernel		7,769
VDH Code (RTP-IN/OUT)		1,141
TCP Interface Modules		
TCPSRV-service routines	1374	
TCPDRI-initialization	764	
TCPNET-VDH interface	971	
TCPELF-EFL calls	1039	
subtotal		4148
TCP user calls (TCPUSR)		1567
Send Process (TCPSND)		1564
Receive Process (TCPRCV)		947
Input Packet Handler (TCPINPKT)		3323
Retransmission Process (TCPRTN)		968
subtotal - TCP system		12,517
FLEA (debugging package)		2,029
EXERCISER (traffic generator/measurement)		1,659
TOTAL		25,115

TCP/ELF Space Requirement

Table 1

System Component		Size in <u>16 bit Words</u>
TCP input co-routine	600	
TCP output co-routine with retransmission	500	
Shared ring buffer	400	
Misc. error handling	600	
TCP-0 subtotal		2,100
Interrupt dispatch (est.)		200
TELNET (est.)		1,500
Terminal I/O (est.)		300
HOST/PRNET software		<u>?</u>
PR Terminal Software total		4,100+ ?

TCP-0 Software Sizes

Table 2

test duration	letter length (in octets)	retransmission time-out (seconds)	no. letters sent	no. of acks sent by retrans. process	no. retransmissions
1 min.	10	0.5	91	93	31
1 min.	20	0.5	92	93	34
1 min.	50	0.5	93	92	31
1 min.	70	0.5	90	93	13
1 min.	10	0.25	106	128	107
1 min.	20	0.25	98	127	93
1 min.	50	0.25	90	124	94
1 min.	70	0.25	99	122	78

Looped Throughput/One Letter at a Time

Table 3

Letter Size (octets)	Packet Retrans. time (sec.)	Retrans. Process Wake UP Time (sec.)	No. Letters		No. Acks Sent by Retrans.		No. Retrans sent		No. Acks only		No. Retrans only		No. ACK and Retrans
			Sent		Retrans.								
-- NO FORCED ACK --													
10	0.5	0.5	92		95		35		65		5		28
10	0.5	0.25	122		124		12		117		6		6
10	0.25	0.25	111		133		70		74		11		59
10	0.25	0.125	97		160		101		88		45		56
-- FORCED ACK --													
10	0.5	0.5	151		0		8		0		8		0
10	0.5	0.25	152		2		3		2		3		0
10	0.25	0.25	165		36		95		12		71		48
10	0.25	0.125	92		58		100		28		70		30
70	0.5	0.25	138		4		14		4		14		0

Forced and Non-Forced ACK Statistics

Table 4

<u>Test Duration</u>	<u>Letter Length (octets)</u>	<u>Retransmission Timeout</u>	<u>No. Letters Sent</u>	<u>No. ACKs Sent by Retransmission Process</u>	<u>No. Retransmissions</u>
2 min.	20	0.5	361	130	114
2 min.	20	0.25	190	190	384(!)
1 min.	20	0.5	180	65	57
1 min.	20	0.25	95	95	192

Equivalent 1 Minute Statistics:

Two Letters Outstanding/Looped Throughput

Table 5

DISTRIBUTION

ARPA

Director (2 copies)
ATTN: Program Management
Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

ARPA/IPT
1400 Wilson Boulevard
Arlington, VA 22209

Dr. Robert Kahn
Mr. Steven Walker

Bell Laboratories

Dr. Elliot N. Pinson, Head
Computer Systems Research Dept.
Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey 07974

Dr. Samuel P. Morgan, Director
Computing Science Research
Bell Laboratories
610 Mountain Avenue
Murray Hill, New Jersey 07974

Dr. C. S. Roberts, Head
The Interactive Computer Systems
Research Department
Bell Laboratories
Holmdel, New Jersey 07733

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02138

Mr. Jerry D. Burchfiel
Mr. R. Clements
Mr. A. McKenzie
Mr. J. McQuillan
Mr. R. Tomlinson
Mr. D. Walden

Burroughs Corporation

Dr. Wayne T. Wilner, Manager
Burroughs Corporation
3978 Sorrento Valley Boulevard
San Diego, CA 92121

Mr. David H. Dahm
Burroughs Corporation
Burroughs Place
P. O. Box 418
Detroit, MI 48232

Mr. B. A. Creech, Manager
New Product Development
Burroughs Corporation
460 Sierra Madre Villa
Pasadena, CA 91109

Cabledata Associates

Mr. Paul Baran
Cabledata Associates, Inc.
701 Welch Road
Palo Alto, CA 94304

California, University - Irvine

Prof. David J. Farber
University of California
Irvine, CA 92664

California, University - Los Angeles

Professor Gerald Estrin
Computer Sciences Department
School of Engineering and Applied Science
Los Angeles, CA 90024

Professor Leonard Kleinrock
University of California
3732 Boelter Hall
Los Angeles, CA 90024

Mr. William E. Naylor
University of California
3804-D Boelter Hall
Los Angeles, CA 90024

Collins Radio Group
1200 N. Alma Road
Richardson, Texas 75080

Mr. Don Heaton
Mr. Frederic Weigl

Defense Communications Engineering
Center

Dr. Harry Helm
DCEC, R-520
1860 Wiehle Avenue
Reston, VA 222090

General Electric

Dr. Richard L. Shuey
General Electric Research
and Development Center
P. O. Box 8
Schenectady, New York 12301

Dr. A. Bell Isle
General Electric Company
Electronics Laboratory
Electronics Park
Syracuse, New York 13201

Mr. Ronald S. Taylor
General Electric Company
175 Curtner Avenue
San Jose, CA 95125

General Motors Corporation
Computer Science Department
General Motors Research Laboratories
General Motors Technical Center
Warren, MI 48090

Dr. George C. Dodd, Assistant Head
Mr. Fred Krull, Supervisory Research
Engineer
Mr. John Boyse, Associate Senior
Research Engineer

Hawaii, University of
The ALOHA System
2540 Dole Street, Holmes 486
Honolulu, Hawaii 96822

Professor Norman Abramson
Professor Franklin F. Kuo

Hughes Aircraft Company

Mr. Knut S. Kongelbeck, Staff Engr.
Hughes Aircraft Company
8430 Fallbrook Avenue
Canoga Park, CA 91304

Mr. Allan J. Stone
Hughes Aircraft Corporation
Bldg. 150 M.S. A 222
P. O. Box 90515
Los Angeles, CA 90009

Hughes Aircraft Company
Attn: B. W. Campbell 6/E110
Company Technical Document Center
Centinela and Teale Streets
Culver City, CA 90230

IBM

Dr. Patrick Mantey, Manager
User Oriented Systems
International Business Machines Corp.
K54-282, Monterey and Cottle Roads
San Jose, CA 95193

Dr. Leonard Y. Liu, Manager
Computer Science
International Business Machines Corp.
K51-282, Monterey and Cottle Roads
San Jose, CA 95193

Mr. Harry Reinstein
International Business Machines Corp.
1501 California Avenue
Palo Alto, Ca 94303

Illinois, University of

Mr. John D. Day
University of Illinois
Center for Advanced Computation
114 Advanced Computation Bldg.
Urbana, Illinois 61801

Institut de Recherches d'Informatique et
d'Automatique (IRIA)
Reseau Cyclade.
78150 Rocquencourt
France

Mr. Louis Pouzin
Mr. Herbert Zimmerman

Information Sciences Institute,
University of Southern California
4676 Admiralty Way
Marina Del Rey, CA 90291

Dr. Marty J. Cohen
Mr. Steven D. Crocker
Dr. Steve Kimbleton
Mr. Keith Uncapher

London, University College

Professor Peter Kirstein
UCL
Department of Statistics &
Computer Science
43 Gordon Square
London WC1H 0PD, England

Massachusetts Institute of Technology

Dr. J. C. R. Licklider
MIT
Project MAC - RTD
545 Technology Square
Cambridge, Massachusetts 02139

MITRE Corporation

Mr. Michael A. Padlipsky
MITRE Corporation
1820 Dolley Madison Blvd.
Westgate Research Park
McLean, VA 22101

Network Analysis Corporation
Beechwood, Old Tappan Road
Glen Cove, New York 11542

Mr. Wushow Chou
Mr. Frank Howard

National Bureau of Standards

Mr. Robert P. Blanc
National Bureau of Standards
Institute for Computer Sciences
and Technology
Washington, D. C. 20234

Mr. Ira W. Cotton
National Bureau of Standards
Building 225, Room B216
Washington, D. C. 20234

National Physical Laboratory
Computer Science Division
Teddington, Middlesex, England

Mr. Derek Barber
Dr. Donald Davies
Mr. Roger Scantlebury
Mr. P. Wilkinson

National Security Agency
9800 Savage Road
Ft. Meade, MD 20755

Mr. Dan Edwards
Mr. Ray McFarland

Norwegian Defense Research Establishment
P. O. Box 25
2007 Kjeller, Norway

Mr. Yngvar G. Lundh
Mr. P. Spilling

Oslo, University of

Prof. Dag Belsnes
EDB-Sentret, University of Oslo
Postbox 1059
Blindern, Oslo 3, Norway

Rand Corporation
1700 Main Street
Santa Monica, CA 90406

Mr. S. Gaines
Mr. Carl Sunshine

Rennes, University of

M. Gerard LeLann
Reseau CYCLADES
U.E.R. d'Informatique
B. P. 25A
35031-Rennes-Cedex, France

Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, CA 94025

Ms. E. J. Feinler
Augmentation Research Center

Dr. Jon Postel
Augmentation Research Center

Mr. D. Neilson, Director
Telecommunication Sciences Center

Dr. David Retz
Telecommunication Sciences Center

System Development Corporation

Dr. G. D. Cole
System Development Corporation
2500 Colorado Avenue
Santa Monica, CA 90406

Telenet Communications, Inc.
1666 K Street, NW
Washington, D. C. 20006

Dr. Holger Opderbeck
Dr. Lawrence G. Roberts
Dr. Barry Wessler

Transaction Technology Inc.

Dr. Robert Metcalfe
Director of Technical Planning
Transaction Technology Inc.
10880 Wilshire Blvd.
Los Angeles, CA 90024

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Mr. David Boggs
Dr. William R. Sutherland

STANFORD UNIVERSITY

Digital Systems Laboratory

Mr. Ronald Crane
Mr. Yogen Dalal
Ms. Judith Estrin
Professor Michael Flynn
Mr. Richard Karp
Mr. James Mathis
Mr. Darryl Rubin
Mr. Wayne Warren

Digital Systems Laboratory Distribution

Computer Science Department - 1 copy
Computer Science Library - 2 copies
Digital Systems Laboratory Library - 6 copies
Engineering Library - 2 copies
IEEE Computer Society Repository - 1 copy

Electrical Engineering

Dr. John Linvill